Neu-NoC: A High-efficient Interconnection Network for Accelerated Neuromorphic Systems**

Xiaoxiao Liu[1], Wei Wen[1], Xuehai Qian[2], Hai Li[3], Yiran Chen[3]
[1]University of Pittsburgh, Pittsburgh, USA
[2]University of Southern California, Los Angeles, USA
[3]Duke University, Durham, USA
{xil116, wei.wen}@pitt.edu, xuehai.qian@usc.edu, {hai.li, yiran.chen}@duke.edu

ABSTRACT

A modern neuromorphic acceleration system could consist of hundreds of accelerators, which are often organized through a network-on-chip (NoC). Although the overall computing ability is greatly promoted by a large number of the accelerators, the power consumption and average delay of the NoC itself becomes prominent. In this paper, we first analyze the characteristics of the data traffic in neuromorphic acceleration systems and the bottleneck of the popular NoC designs adopted in such systems. We then propose Neu-NoC – a high-efficient interconnection network to reduce the redundant data traffic in neuromorphic acceleration systems and explore the data transfer ability between adjacent layers. A sophisticated neural network aware mapping algorithm and a multicast transmission scheme are designed to alleviate data traffic congestions without increasing the average transmission distance. Finally, we explore the sparsity characteristics of fully-connected NNs. Simulation results show that compared to the most widely-used Mesh NoC design, Neu-NoC can substantially reduce the average data latency by 28.5% and the energy consumption by 39.2% in accelerated neuromorphic systems.

1. INTRODUCTION

Neural Network (NNs) have been widely utilized in many cognitive applications and achieved remarkable successes in the scenarios where the volume of the data to be processed is beyond the capability of a human being, such as robotics [13], self-driven car [17], etc. To better infer the input data, the scale of the adopted NN is usually very large. As a recent example, Google uses a database of about 30 million moves to teach AlphaGo to play Go and beat human champions [16]. Compared to conventional solutions, various hardware accelerators were recently proposed as an energy-efficient alternative to perform NN computations [4].

Network-on-chip (NoC) is widely used in modern multicore systems for on-chip data transferring [6]. Neuromorphic systems can also benefit from NoC to manage the data movement between the accelerators. However, when the scale of the NN is large, the data traffic in a neuromorphic system can be very busy, making the NoC become the bottleneck of the system performance and energy. For instance, the data communication may count for more than 30% computation cost in a deep learning accelerator and grows up rapidly with the increase of the system scale [4]. Meanwhile, about 26% chip area and more than 10% total energy consumption comes from the NoC [4]. The non-uniformity of the transmission data size over the connections, the redundancy of the data transmission between the NN layers, and the local congestions all significantly degrade the efficacy of the NoC in a neuromorphic acceleration system. In [3], Carrillo *el at.* implemented a star-mesh-based NoC for spiking NNs to replace the conventional shared bus solution. However, considering the constraints of spiking NNs and spiking signals, it is difficult to apply the start-mesh-based NoC onto other NNs that have different topologies. Ring topology has been recently used in the NOC design of some multicore systems for its simplicity, i.e., Intel Nehalem [14]. But its implementation in a large system is doubtful by taking into account the increased hop count and long wire delay. In this work, we propose a hybrid ring-mesh NoC architecture (namely, Neu-NoC) that can adapt to the unique communication patterns in neuromorphic acceleration systems to achieve better performance, less energy, and smaller area. Compared to the prior-arts, our key contributions are:

- We perform ***comprehensive evaluations of traditional NoC in neuromorphic acceleration systems*** to identify design bottlenecks and limitations;
- We design ***a novel hybrid NoC for neuromorphic acceleration systems*** based on the characteristic of NN data traffic, offering significant performance and power efficiency improvement;
- We propose ***a sophisticated NN-aware mapping algorithm*** to reduce the distance of the data communication and alleviate the congestions on neuromorphic acceleration systems as the same time;
- We develop ***a multicast type of data transmission*** to reduce the amount of packets with same data in each routing path;
- We design ***a new type of flit*** to leverage the sparse feature of feature map to further mitigate the data traffic congestion with an acceptable accuracy loss.

A set of popular NN and machine learning applications is adopted in the evaluation of Neu-NoC. Simulation results show that Neu-NoC can reduce the transaction latency up to 28.5% compared to the Mesh NoC baseline and averagely achieve 39.2% energy saving.

2. BACKGROUND AND MOTIVATION

2.1 Neural Networks (NN)

In this work, we mainly focus on Multilayer Perceptron (MLP), which is a feedforward artificial neural network that widely utilized in classification algorithms such as the classification layer in deep neural networks (DNN) [8] and convolutional neural networks (CNN) [15], and approximate computing [9]. MLP presents not only the basic computation patterns of DNN, i.e., matrix multiplication followed by nonlinear activation functions (e.g., sigmoid etc.) at each layer, but also the intensive communications within the layers. Figure 1 depicts the hardware utilization of Alexnet [12] running on Nvidia GeForce GTX TITAN X GPU [1] where

*This work is supported in part by NSF-1725456, NSF-1615475, and DOE DE-SC0018064

978-1-5090-0602-1/18/$31.00 ©2018 IEEE

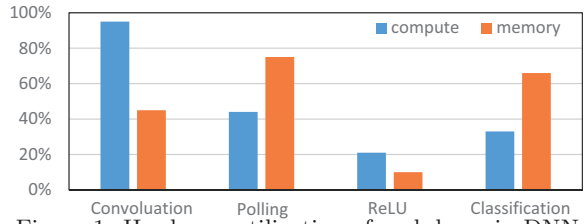


Figure 1: Hardware utilization of each layer in DNN.

MLP processes the largest memory-to-computing ratio. As such a memory-to-computing ratio directly links to the traffic intensity of the NoC on a neuromorphic system, we choose MLP as the target in our study: the performance of MLP on each NoC design reflects the worst-case efficacy of the NoC.

2.2 Neuromorphic Acceleration System

On a neuromorphic acceleration system, data are stored in a distributed manner (i.e., the weights of synapses) and participates in the computation through local neurons. Although these architectures greatly mitigate the requirement of memory bandwidth, long-range connectivity across cores emerges as a new challenge in hardware development. For example, in IBM TrueNorth system, the local neuronal execution within a neurosynaptic core is extremely efficient, while the energy consumption of core-to-core communication increases rapidly with the distance between source and destination [2]: an inter-core data transmission could consume 224× energy of an intra-core one (i.e., $894pJ$ vs. $4pJ$ per spike per hop). As the scale and density of the NN increase, more inter-core interconnections are introduced; the effect of long-range connectivity and communication quickly becomes a severe design challenge. To overcome this challenge in NN acceleration, both hardware and software solutions are explored. The hardware approaches attempt to build a specialized circuit and/or architecture including some customized network models. These new models, however, are often inconsistent with their state-of-the-art version, resulting in low inference accuracy [2]. On the contrary, the software approaches mainly focus on reducing the scale and connectivity of DNN models while still retaining the accuracy [10]. However, implementing such pruned models on hardware can be very challenging due to the scattered memory access patterns.

2.3 Motivation of Our Work

NoC is a critical part in neuromorphic acceleration system designs because the data traffic pattern is significantly different from that in conventional multicore systems. Figure 2 depicts a neuromorphic acceleration system that is composed of 648 processing engines (PEs) [4] connected with a 26×26 Mesh NoC as our baseline. The weights are stored

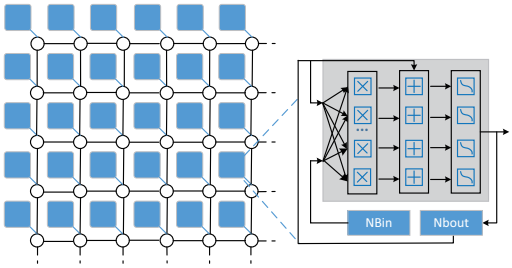


Figure 2: Baseline design of a neuromorphic acceleration system with NoC.

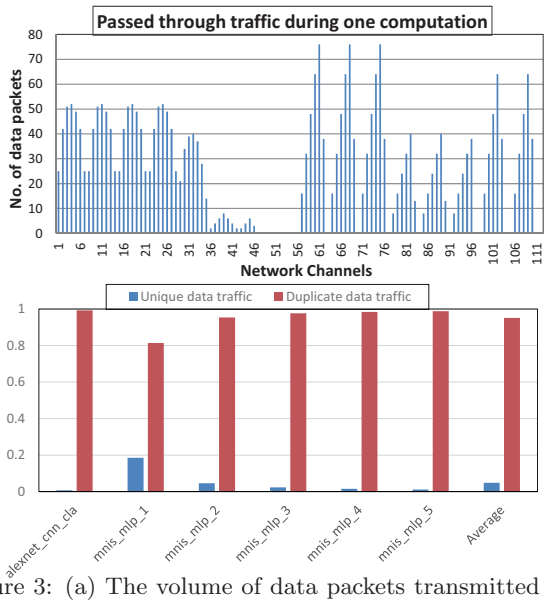


Figure 3: (a) The volume of data packets transmitted over different channels; (b) The ratio of duplicated data packets.

in distributed local memories and fetched through a specific high-throughput memory bus within a short distance. The matching relations between nueron outputs and weights are stored as the source address information in head flit and also the sequential order of flits in a packet.

In MLP, the neurons in one layer only communicate with the neurons in the next adjacent layer. As an initial study, we stimulate the data traffic of 6 selected NN benchmarks running on the NoC of the neuromorphic acceleration system depicted in Figure 2. More details about these bencharmks can be found in Table 1 in Section 4. Figure 3(a) shows the result of running mnist_mlp_1 that on average, 57.6% of the total data packets travel through only 29% of total channels during the NN computation. The data traffic is particularly concentrated on the transmission channels that connecting two adjacent layers while the channels between the neurons in one layer are idle most of the time. Obviously such communication pattern leads to very unbalanced traffic patterns and causes congestions over the NoC.

In addition, every neuron in a layer of the NN sends the same value to its connected neurons in the next layer. It means that a node of a NoC could send multiple packets containing the same data to a set of nodes. We compare the number of the packets with unique data and the number of the packets with the same data but sent to different destination nodes. The results in Figure 3(b) show that a significant amount of packets are indeed used to deliver the same data to different nodes. Such traffic pattern is rare in a conventional computing model and offers a great improvement opportunity for the NoC design in neuromorphic acceleration systems.

3. IMPLEMENTATION OF NEU-NOC

3.1 Hierarchical Structure of Neu-NoC

Our initial analysis of the traffic patterns on traditional mesh NoC in neuromorphic systems inspired us to propose Neu-NoC – an efficient NoC architecture that can suppress unnecessary data transfers of the same data and reduce the bandwidth consumption by consolidating neurons on the

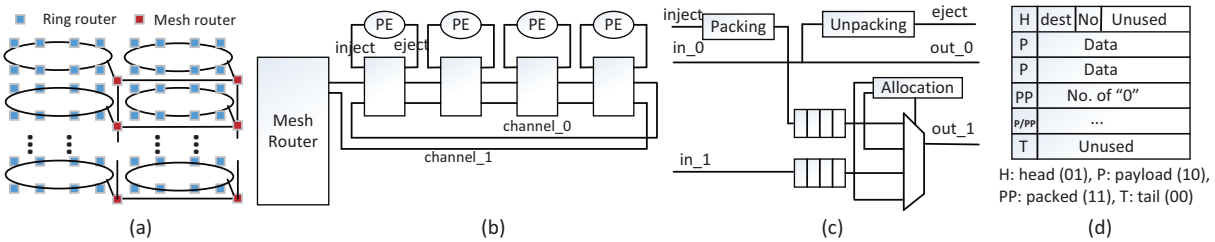


Figure 4: (a) Hierarchical Neu-NoC; (b) ring bus; (c) ring router; (d) packet format.

same NN layer into local nodes.

Figure 4 (a) presents the overview of Neu-NoC architecture. The topology of Neu-NoC is a hybrid ring-mesh NoC structure. It consists of local rings and a global mesh that interconnects all the local rings. Figure 4 (a) shows a configuration example with 8 PEs connected in one ring bus which is communicated to others through a Mesh NoC. In Neu-NoC, we use rings to cluster the neurons in the same layer to reduce the number of the data packets contenting the same data that need to be transferred: the neurons connected by one ring only send one copy of their data to the rings that connect the neurons in the next layer.

The local ring topology consists of two channels – a channel to receive data and a channel to pass the neuron output to the next layer, as shown in Figure 4(b). As a result, the output transferring of the neurons does not need to wait until the ring is idle. The traffic stalls are greatly reduced. Neu-NoC consists of two types of routers – a ring router connected to the PEs in each local ring and a mesh router connected to each local ring and other four mesh routers on its four neighbour directions. A block diagram of the router's microarchitecture is shown in Figure 4(c). The ring router consists of a 2-to-1 multiplexers, buffers, and function blocks for pack/unpacking data packets. Additionally, the ring router supports the arbitration with different priorities in *Allocation* function block, i.e., data packets in-flight always have a higher priority than the newly-injected packets. We adopt the typical mesh router design with wormwhole flit-based flow control [6] for the global mesh network to maintain high flexibility of mapping different NN topologies.

3.2 NN-aware NoC Mapping

3.2.1 *Effect of NoC mapping*

As the data transmission requirement (e.g., the source and destination neurons, amount of data) in a NN will not change during the computation, the data routing path is decided by NN-to-NoC mapping and routing algorithm [6]. However, normal direct-mapping or random-mapping may not be optimal: the distance between two connected neurons and the congestion ratio in each NoC channel are more

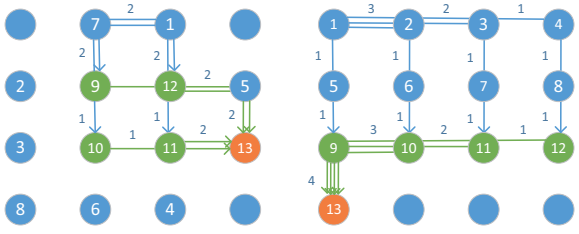


Figure 5: Different PE group placement in Neu-NoC for `mnist_mlp_2` (MLP topology after mapping: 8-4-1).

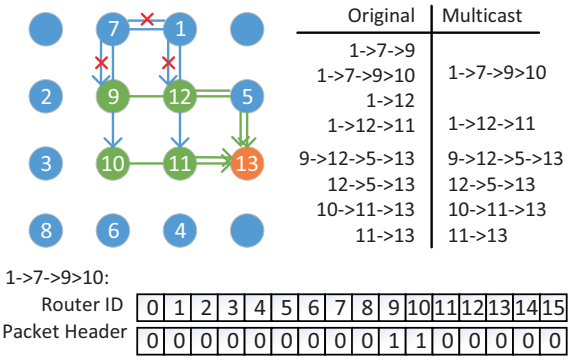


Figure 6: Multicast and Packet Header Example.

likely to enlarge the transmission latency.

Here we use an example to illustrate how different mapping schemes influences the average hop count and the congestion of the data traffic. Fig. 5 shows the comparison between a random mapping and our proposed NN-aware mapping based on the NN topology. Here each circle represents a local ring network and the number in each circle labels the NN layers that the ring network resides on. The arrows show the path between routers, and the number on each arrow represents how many packets pass through the path at a moment of MLP computation. In direct mapping scheme, the layers are placed sequentially on the NoC, which may lead to a very high number of hops between the PEs in the adjacent layers. Our NN-ware mapping scheme, however, allows more PEs to access the PEs in adjacent layers with fewer hops.

To obtain the minimum hop counts and a more balanced NoC, we design a NN-aware mapping which leads to the mapping solution with significantly reduced routing distance and distributed data traffic among the NoC for obtaining latency reduction.

3.2.2 *Problem formulation and definition*

The problem of mapping a NN onto a NoC is a NP problem. We introduce the following definitions to help to formulate the problem that our proposed NN-aware mapping algorithm is facing to find the optimal NoC mapping solution of the NN:

Definition 1: A *Neural Network Communication Graph* (NNCG) is a directed graph denoted by $G(N, A)$, in which each vertex n_i represents one neuron in NN, and each directed arch $a_{i,j}$ models a communication flow from one neuron n_i to one of its connected neurons n_j in the next layer.

Definition 2: An *architecture characterization graph* (ARCG) $G'(U, L)$ is also a directed graph that represents the physical NoC, where each vertex u_i denotes a node in the NoC and each edge l_i represents a physical link.

Definition 3: For an ARCG $G(U, L)$, a deterministic routing function $\Re : R \rightarrow P$ maps $r_{i,j}$ to one routing path

Algorithm 1 NN-aware mapping algorithm

procedure NN-AWARE-MAPPING
input:
  $NNCG(N,A) \leftarrow$ NN communication graph
  $ARCG(U,L) \leftarrow$ architecture characterization graph
output:
  $map \leftarrow$ from$NNCG(N,A)$to$ARCG(U,L)$
  for $n_i \in NNCG(N,A)$ **do**
    map n_i to $u_x \in ARCG(U,L)$
    Generate map$(n_i) \in U$
    for each $a_{i,j}$ in A **do**
      $H \leftarrow \sum |row_i - row_j| + |col_i - col_j|$
      if $H \leq H_{min}$ **then**
        $H_{min} \leftarrow H$
        $map_{min} \leftarrow map(n_i)$
  for each $p_{i,j} \in P_{Hmin_{i,j}}$ **do**
    $l_{traffic} \leftarrow l_i + \cdots + l_j, l_i, \ldots, l_j \in p_{i,j}$
    if $l_{traffic} \geq l_{max}$ **then**
      $l_{max} \leftarrow l_{traffic}$
      recode$l_{max} foreach map(n_i)$
output $map(n_i)$ with smallest l_{max}

$p_{i,j}$, where $p_{i,j} \in P_{i,j}$.

Based on these definitions, the problem of the NN-aware mapping can be formulated as follow: Give an NNCG and an ARCG, we need to find a mapping function $map()$ which satisfies:

$$min\{T_{NoC} = \frac{\sum_{y=1}^{Q}\sum_{x=1}^{P} |row_{n+1,y} - row_{n,x}| + |col_{n+1,y} - col_{n,x}|}{M^2(M-1)} + Pack_{sum}\} \quad (1)$$

We assume that the row and the column numbers of the x^{th} PE are $row_{n,x}$ and $col_{n,x}$, respectively. Correspondingly, $row_{n+1,y}$ and $col_{n+1,y}$ are the row and the column numbers of the y^{th} connected PE in the adjacent layer.

such that:

$$3\forall n_i \in N, map(u_i) \in U \quad (2)$$

$$\forall n_i \neq n_j \in N, map(u_i) \neq map(u_j) \quad (3)$$

$$\forall a_{i,j} \in A, p_{i,j} \in P_{i,j} \quad (4)$$

The proposed algorithm is shown in Alogrithm 1. The results of the hop count and max data load of the selected 7 benchmarks using the proposed NN-aware mapping algorithm are summarized in Table 1 and discussed in Section 5.

3.3 Multicast Transmission

Since each neuron sends the same data to all its connected neurons in the next layer and some of the data packets share the same routing path, we design a multicast type of data transmission to combine the data packets from the same source node into one data packet, which will take same path during routing. For simplicity of the hardware, only the data packets that are completely contained by a other packet will be merged to the latter one. Table 1 shows the total number of the hops that all the data packets pass through in one NN computation before and after multicasting is applied.

To support the multicast transmission, we redesign the header of packet by introducing a bit string encoding scheme [5] to carry multiple destination nodes and add a decoding and bit reset function in the router, as shown in Figure 6. In bit string encoding, each destination can be represented by only one bit. Figure 6 depicts an example of a multicast and its corresponding packet header. The length of the encode bit string in the header equals the number of nodes in the Mesh NoC; each bit represents a node and setting the bit to 1'b1 means that the node is one of the destinations. When a packet arrives at one of its destination, the bit corresponding to the destination will be reset to 1'b0. Among all the selected benchmarks, the longest routing path is 16 links, which exists in *alexnet_cnn_cla*.

3.4 Sparsity-aware Traffic Reduction

As many prior-arts have proven, the highly abstracted feature map in DNN can be very sparse [7]. This sparsity can also help to reduce the traffic on the NoC. We introduce a new type of flit, namely, all-zero flit, to present a sequence of 0's in the data. The all-zero flit can be sent in any order between the head flit and the tail flit in a packet. Such a scheme allow only one flit indicate multiple sequential flits of 0's in the original flit designs: We add one bit as a flag to denote if the packet is packed with all zero data, and use the payload area to denote the number of the continuous 0's. We implement a function block at each input/output port that connecting the mesh and the local ring to pack/unpack all-zero flits, as shown in Figure 4. In the packing block, we use a counter to detect the sequence of 0's and a "pack" signal will be asserted if packing is needed. Similarly, in the unpacking block, as soon as an all-zero flit is detected, the router will unpack the all-zero flits and restore the sequence of 0's. The packing/unpacking blocks only introduce 3.1% area overhead to the router design.

Figure 4(d) depicts the format of a data packet. We add a new flit type - 'PP' to represent the all-zero flit in which the 2-bit head is 2'b11 and the "body" gives the number of 0's. The formats of the other flits are similar to that in conventional wormhole flit-based flow control NoC.

4. EXPERIMENTAL METHODOLOGY

In our experiments, we use MLP on MNIST database and the classification layer of AlexNet on ImageNet as our NN examples. MLPs on MNIST are trained without data augmentation and AlexNet [12] is trained using Caffe [1]. During the forwarding of the NNs, we zero out the activations propagated to the hidden layers when their absolute values are smaller than a predetermined threshold. We define the accuracy as the *mean square error* (MSE) between the actual and target outputs under the training vectors. Table 1 gives the benchmark information such as the implementation details, the initial training accuracy, and other informations after feature maps are pruned.

We modify *Booksim* [11] – a cycle-accurate NoC simulator, by adding NN types of traffic module to mimic the data transfer in neuromorphic acceleration systems during the operations. Each Node is assumed to be a processing engine (PE) that has the design and computing ability similar to the one in [4]. A constant delay is added as the calculation delay before the output packets are generated when all the input data from the previous layer are collected. Table 2 summarizes the parameters of our simulation platforms. The

[1] `https://github.com/BVLC/caffe/tree/master/models/bvlc_alexnet`

Table 1: The description and implementation details of the selected benchmarks (Ran: Random, Mult: Multicast)

Benchmark	Topology	mapping in PEs	NoC		Initial	Zero flit	Hops No.		Max load		Packet No.	
			Mesh	Neu	accuracy	traffic	Ran	NN	Ran	NN	NN	Mult
mnist_mlp_1	300-100-10	19-7-1	6x6	3x3	98%	63.8%	12	4	3	1	-	-
mnist_mlp_2	1000-500-10	63-32-1	10x10	4x4	98.27%	55.8%	124	76	16	4	36	20
mnist_mlp_3	1500-1000-500-10	94-63-32-1	14x14	5x5	98.28%	58.3%	550	421	24	14	132	87
mnist_mlp_4	2000-1500-1000-500-10	125-94-63-32-1	18x18	7x7	98.32%	53.2%	1774	1308	36	11	324	224
mnist_mlp_5	2500/2000/1500/1000/500/10	157-125-94-63-32-1	22x22	8x8	98.22%	56.8%	3958	3074	27	16	644	419
alexnet_cnn_cla	4096-4096-1000	256-256-63	24x24	9x9	56.60%	48.3%	8218	7024	90	54	1260	563

Table 2: The system simulation configuration

PE	256 16-bit multipliers, 256 16-bit adders
Mesh	26×26 mesh topology, XY routing, wormhole switching, 4-stage pipeline, 4 virtual channels per port
Fattree_Mesh [4]	4-ary fat-tree with a height of 3 in local, 6×6 mesh in global connection, nearest-common ancestor routing for fat-tree, XY-routing for mesh
Neu_NoC	8 nodes in local ring topology, 9×9 mesh in global with XY-routing, 1 cycle router delay for passing through, min 3 cycle router delay for inject/eject

energy consumption of the NoC is estimated by also *Booksim* with 45nm technology.

5. EXPERIMENTAL RESULTS

5.1 Impact of Concentration Degree

Allocating more PEs on each ring network not only reduces the redundant data packets sent to the PEs of the next NN layer but also reduces the latency and energy overhead of the routing on the global mesh. However, increasing the length of the ring network will also increase the wire delay. Figure 7 compares the execution times of all the NN benchmarks using a Neu-NoC with ring and mesh configurations. Here the total number of the PE is set to 648. When the number of PEs located in the same ring increases from 1 to 8, the NoC performance keeps improving due to the reduction of the global hops count. However, such trend stops when the scale of the ring network is too large (i.e., 64 PEs) so that the increase of the local wire delay has surpassed the reduction of the global hops count. Among all the tested benchmarks, `alexnet_cnn_cla` is most sensitive to the scale of the ring network because its large hidden layers (4096 neurons) fully occupies all the PEs in each ring. In the following experiments, we choose 8 PEs as our default configuration which demonstrates the best balance between the local wire latency and global hops count.

5.2 Impact of Feature Map Sparsity

We also evaluate the impact of the NN sparsity on the efficacy of Neu-NoC. Here the sparsity is defined as the percentage of 0's in the feature map. It is known that increasing the sparsity will degrade the accuracy of the NN. However, as shown in Figure 8, the accuracy degradation of all the benchmarks can be maintained at a very low level even the corresponding NN sparsity is very high: In *alexnet_cnn_cla*, which is the worst case, the average sparsity is 78.6% across three feature maps while the incurred accuracy is less than 1%. In Neu-NoC, we can dynamically sparsify the NN by zeroing out any features smaller than 1%. The column of "Zero flit" of Table 1 shows the ratios between the corresponding traffic and the baseline traffic.

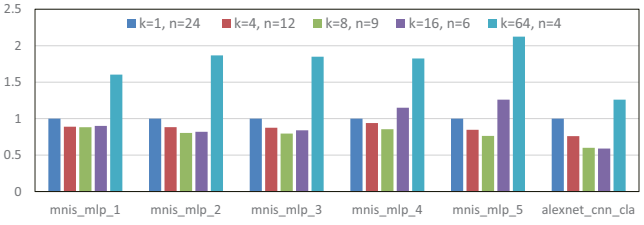


Figure 7: Impact of concentration degree (k: the number of the nodes in ring; n: the numbers of columns/rows of mesh).

5.3 Effectiveness of NN-aware Mapping

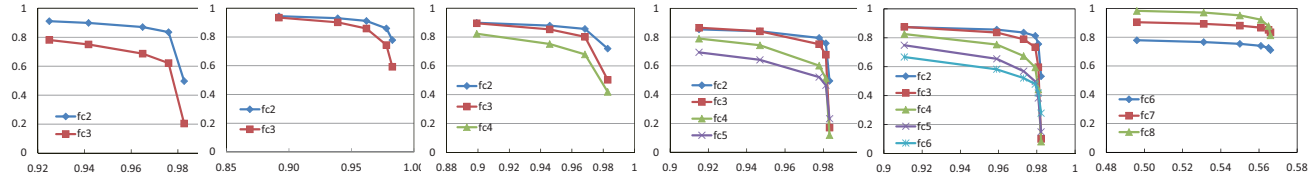
Figure 9 illustrates the impacts of different NN mapping schemes on the average latency of packet transmission in Neu-NoC in the simulated neuromorphic acceleration system. Here $x-$axis is the data injection rate of the input neurons of the first layer, which is defined as the number of packets are input in each node in every cycle. The injection rate between the NN layers is triggered after the neuron collects all the inputs from the previous layer. As shown in Figure 9, NN-aware mapping always achieves the best performance in all 6 tested benchmarks as well as the best tolerance to the traffic load (data injection rate) increase. For example, in 5 out of 6 benchmarks, the average packet latency of NN-mapping maintains low until the data injection rate exceeds about 0.09 while that of the other two mapping schemes starts to rockets when the data injection rate reaches about 0.05. In addition, NN-aware mapping demonstrates great scalability by showing more significant average packet latency reduction when the network scale is large, e.g., in `alexnet_cnn_cla` (see Figure 9(f)).

5.4 Effectiveness of Multicast

Figure 10 compares the average packet latencies before and after applying multicast in 5 benchmarks: `mnist_mlp_2` ∼ `mnist_mlp_5` and `alexnet_cnn_cla`. Note that here we did not include `mnist_mlp_1` because `mnist_mlp_1` does not have any data packet can be represented by other packets which share the same source node and contain its routing path. The result shows multicast successfully reduces that the average packet latency in all 5 benchmarks, especially in `Alexnet_cnn_cla` and `mnist_mlp_2` which have less layers and hence, less complicated mapping and routing paths than `mnist_mlp_3` ∼ `mnist_mlp_5`. The packet counts before and after applying multicast are depicted in Table 1.

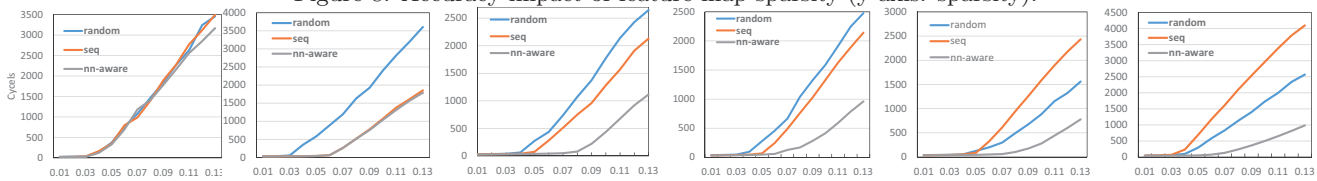
5.5 Evaluation of Neu-NoC

We compare the execution time and energy consumption of three NoC designs: 2D Mesh, Fattree_Mesh [4], and Neu-NoC (including the design with and without zero filt design) for the 6 benchmarks, as depicted in Figure 11. All the results have been normalized to the baseline Mesh NoC design. Compared to Mesh NoC, Neu-NoC averagely reduce the average packet latency and energy consumption of the NoC by 23.2% and 31.1%, respectively. Compared to Fattree-Mesh NoC [4] which has been used in a neuromorphic acceleration system, Neu-NoC achieves on average 6% average packet latency reduction and 13% energy consumption saving, respectively. If we allow 1% accuracy degradation of NN



(a) mnist_mlp_1 (b) mnist_mlp_2 (c) mnist_mlp_3 (d) mnist_mlp_4 (e) mnist_mlp_5 (f) alexnet_cnn_cla

Figure 8: Accuracy impact of feature map sparsity (y-axis: sparsity).



(a) mnist_mlp_1 (b) mnist_mlp_2 (c) mnist_mlp_3 (d) mnist_mlp_4 (e) mnist_mlp_5 (f) alexnet_cnn_cla

Figure 9: Average packet latency of different mappings (x-axis: injection rate).

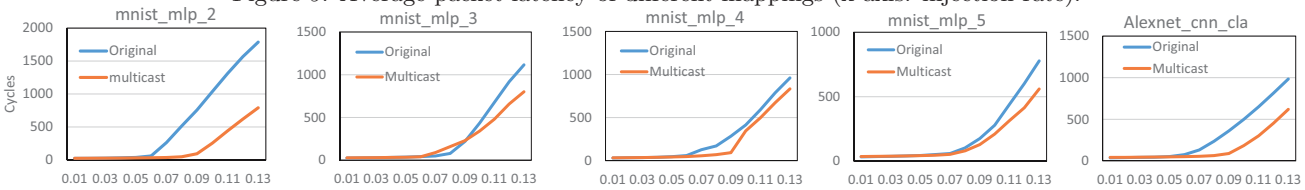


Figure 10: Average packet latency of before and after applying multicast (x-axis: injection rate).

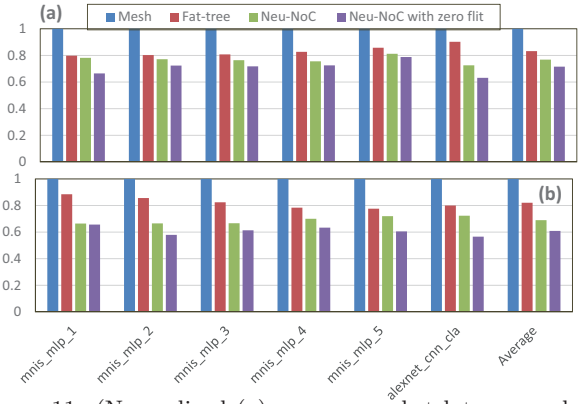


Figure 11: (Normalized (a) average packet latency and (b) energy of all the NoC designs.

(See Table 1), Neu-NoC with zero filt design can further reduce the average packet latency and energy consumption by 11.7% and 21.19%, respectively, compared to Fattree-Mesh.

6. CONCLUSION & FUTURE WORKS

In this work, we propose Neu-NoC, a novel NoC design to optimize the traffic in neuromorphic acceleration systems. A set of techniques, i.e., NN-aware mapping, multicast, and sparsity-aware traffic reduction, are proposed to reduce average hops account and redundant traffics. Our results show that Neu-NoC can effectively reduce the average packet latency and energy consumption in the tested 6 MLP benchmarks by on average 23.2% and 31.1%, without incurring any accuracy degradations. Moreover, slightly relaxing the accuracy requirement of the benchmarks by 1% can further save the average packet latency and energy consumption by 28.5% and 39.2%, respectively. Our future work will focus on applying Neu-NoC to accelerate other components of DNNs such as convolutional and pooling layers.

7. REFERENCES

[1] "Gtx titan x," http://www.geforce.com/hardware/10series/titan-x-pascal.

[2] F. Akopyan, *et al.*, "Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip," *ICS*, vol. 34, no. 10, pp. 1537–1557, 2015.

[3] S. Carrillo *et al.*, "Scalable hierarchical network-on-chip architecture for spiking neural network hardware implementations."

[4] Y. Chen *et al.*, "Dadiannao: A machine-learning supercomputer," in *MICRO*, 2014, pp. 609–622.

[5] C. Chiang and L. Ni, "Multi-address encoding for multicast," *PCRC*, pp. 146–160, 1994.

[6] W. Dally *et al.*, *Principles and practices of interconnection networks*. Morgan Kaufmann Inc, 2004.

[7] R. Girshick *et al.*, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *CVPR*, June 2014.

[8] G. Hinton *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *Signal Processing*, vol. 29, no. 6, pp. 82–97, 2012.

[9] K. Hornik *et al.*, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[10] F. Iandola *et al.*, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016.

[11] N. Jiang *et al.*, "A detailed and flexible cycle-accurate network-on-chip simulator," in *ISPASS*, 2013, pp. 86–96.

[12] A. Krizhevsky *et al.*, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012, pp. 1097–1105.

[13] R.-J. Lian, "Adaptive self-organizing fuzzy sliding-mode radial basis-function neural-network controller for robotic systems," *IE*, vol. 61, no. 3, pp. 1493–1503, 2014.

[14] C. Park *et al.*, "A 1.2 tb/s on-chip ring interconnect for 45nm 8-core enterprise xeon® processor," in *ISSCC*, 2010.

[15] T. Sainath *et al.*, "Deep convolutional neural networks for lvcsr," in *ASSP*, 2013, pp. 8614–8618.

[16] D. Silver *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[17] C. Szegedy *et al.*, "Object detection using deep neural networks," 2016, uS Patent 9,275,308.